UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/824,282	04/14/2004	Mark L. Roth	5681-75900	6128

58467 7590 10/02/2008

MHKKG/SUN
P.O. BOX 398
AUSTIN, TX 78767

EXAMINER	
VU, TUAN A	
ART UNIT	PAPER NUMBER
2193	
MAIL DATE	DELIVERY MODE
10/02/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/824,282	ROTH, MARK L.	
	Examiner	Art Unit	
	TUAN A. VU	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☐ Responsive to communication(s) filed on *29 August 2008*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-5,7,8,10-19,21,23-30,32 and 34-36* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-5, 7-8, 10-19, 21, 23-30, 32, 34-36* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 8/29/08.

As indicated in Applicant's response, claims 1, 15, 26 have been amended, and claims 6, 9, 20, 22, 31, 33 canceled. Claims 1-5, 7-8, 10-19, 21, 23-30, 32, 34-36 are pending in the office action.

Claim Objections

2. Claims 1 is objected to because of the following informalities: the language (last paragraph of claim) recited as "... configured to generate *one or more output respective software documentation sets of types* to which the plug-in corresponds" appears to have a typographical omission and amounts to a hard-to-construe syntax as identified from above (i.e. the italicized phrase). This phraseology from above will be treated as though the output is one or more documentation sets based on the type to which the plug-in corresponds. Appropriate correction is required.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 1-2, 4-5, 7-8, 10-12, 14-16, 18-19, 21, 23, 25-27, 29-30, 32, 34, 36 are rejected under 35 U.S.C. 103(a) as being unpatentable over Clausen et al., USPN: 6,732,330 and further in view of Murren et al, USPN: 7,000,185 (hereinafter Murren).

As per claim 1, Claussen discloses a system, comprising:

a processor; and memory coupled to the processor and configured to store program instructions executable by the processor (col. 26 lines 8-29) to implement a software documentation generator configured to:

input a plurality of sources related to a software program, wherein the plurality of sources comprises a plurality of different types of sources (scripting languages - col. 2 lines 58-65; multiple scripting languages col. 3 lines 12-16) and comprises one or more of a software library documentation file (e.g. *custom tags ... tag libraries...tab library xml file* - col. 5 line 50-66; taglib – col. 7 lines 17-41) for the software program and software program source code for the software program e.g. col. 6 lines 1-21 – Note: tag libs for JSP construct reads on library file for source program of type JSP or XML – see col. 5 line 50-6);

analyze the one or more sources to identify a type of each of the sources (e.g. step 308 – Fig. 3);

extract information from the plurality of sources based on the type of the source (e.g. Fig 3, 4), wherein the software documentation generator includes a input source plug-in is configured to extract information (e.g. Fig. 4 steps 419-422 – Note: DOM or tagbean APIs p– col. 7 lines 64 to col. 8 lines 14; Fig. 7 and related text - and custom tag component to support parsing of DOM tree reads on input source plug-ins) from sources of a type to which the plug-in corresponds (step 422, jsp-directive page – Fig. 4 Note: plug in to register tag lib for a corresponding JSP element on the DOM reads on plug-in corresponding to type);

aggregate the extracted information into a uniform format (e.g. registered through an XML tag-library-xml – col. 5 lines 52-54; step 312 – Fig. 3; DOM data structure - col. 16 line 35

to col. 17 lines 20 - Note: DOM and XML format reads on uniform aggregate reusable for generating new script files); and

transform the aggregated information into one or more specified sets of software documentation (Fig. 5; *multiple scripting language ... another language* - col 10 lines 2-14) for the software program, wherein the software documentation generator includes a transformer plug-in set corresponds to one or more respective types of output software documentation sets (step 510, Fig. 5 - Note: java bean and XSL templates or stylesheet -col. 25 lines 49-61;*tagbean.process* - Fig. 6 - reads on transformer plug-in to create *new document* – see step 618, Fig. 6) and is configured to generate one or more output respective software documentation sets of types to which the plug-in corresponds (e.g. *into a script code by a Java object or an XSL style sheet* – col. 49-61).

Claussen does not explicitly disclose wherein the software documentation generator includes a plurality of different input source plug-ins, wherein each input source plug-in corresponds to a respective one of the source types, nor does Claussen discloses a plurality of different transformer plug-in sets, wherein each transformer plug-in set corresponds to one or more respective types of output software documentation sets and each transformer plug-in set is configured to generate one or more output respective software documentation sets of types to which the plug-in corresponds. But Claussen discloses using DOM api to parse elements of DOM tree (col. 8 lines 46-65), tagbean and tag lib API (tag-library-xml – col. 5 lines 52-54) to register DOM elements in XML form (col. 9 li. 22-61), applying XSL to validate XML element parsed from a DOM tree (stylesheet 508 -Fig. 5) and appropriate Java bean to write a corresponding script construct output (step 616, 618 – Fig. 6); hence the relationship between the

type of source file (XML/DOM tag) and the pertinent type of plug-ins (e.g. bean for XML) needed is implied. Analogous to the DOM parsing by Claussen, Murren discloses handling of source input to extract tag information based on JSP type, or HTML type (Fig. 2) in order to select the proper APIs methods to address the attribute (Fig. 7) of the parsed source thereby yield a output HTML file (Fig. 8). Based on the concept to using API plug-ins and the application of proper APIs in handling of type of the parsed source as set forth in Murren from above, it would have been obvious for one skill in the art at the time the invention was made to implement the plugins by Murren (*plug-ins* – col. 4 lines 7-19, lines 28-40) so that plug-ins for handling a source and to generate output document exist in plurality thereof where each is instantiated for the corresponding type as evidenced from above because this would obviate writing APIs from scratch or downloading or compiling code, in alleviating undue exertion of effort and resources during browser application runtime as purported above (see Claussen: plug-ins col. 4 lines 28-40) whereby plug-ins usage would be a known practice for its being expedient usability.

As per claim 2, Claussen discloses wherein one of the sources comprises the software library documentation file (e.g. tag libraries, taglib, support for JSP 1.0 mechanism – col. 3 lines 22-66 – Note: library file with purpose to support a specific JSP reads on documentation support file).

As per claim 4, Claussen discloses wherein one of the sources comprises the software program source code (e.g. web page file – col. 44-65).

As per claim 5, Claussen discloses wherein the documentation sets comprise documentation for one or more application programming interfaces (API) provided by a software library (e.g. jspServicemethodDefinition , methodDefinition – col. 29-39 – Note: definition

generated during invoking of *tagbean process* – Fig. 6 - and related Jsp server API reads on documentation sets for java APIs based on taglibs).

As per claim 7, Claussen discloses wherein an input source plug-in is configured to generate information not included (e.g. parent document – col. 7 lines 53-62; step 506, Fig. 5; *CDATA around any text* 314 – Fig. 3) in the corresponding source file.

As per claim 8, Claussen discloses wherein each input source plug-in is configured to output data in the uniform aggregate format (e.g. step 204, Fig. 2; step 312, Fig. 3; step 706 Fig. 7 – Note: standardized multi-platform neutral meta-language in XML reads on aggregate uniform format).

As per claims 10-11, Claussen discloses wherein the input source plug-ins are configured to produce a uniformly formatted aggregate input document (refer to claim 8) and wherein each transformer plug-in set is configured to input data included in the uniformly formatted aggregate input document (step 502, Fig 5; col. 5 line 42 to col. 6, line 28 – Note: taglib based on Dom and XML construct and identifying of stylesheet for that construct in rendering a output reads on transformer configured to input XML for a transforming process – Fig. 5); wherein a transformer plug-in set is configured to generate information not included (e.g. *dtd = currentTime.dtd … jspServicemethodDefinition* -col. 6 lines 12-43) in the uniformly formatted aggregate input document.

As per claim 12, 14, Claussen discloses wherein the output software documentation sets comprise one or more text files and wherein the output software documentation sets comprise one or more hypertext markup language (HTML) files. (e.g. step 510 – Fig. 5; col. 10 lines 2-7; *HTTPResponse* – col. 5 lines 5-19).

As per claim 15, Claussen discloses a method, comprising

receiving information from multiple sources related to a software program, wherein the multiple sources comprise a plurality of different types of sources (refer to claim 1) and comprise one or more of a software library documentation file (refer to claim 1) for the software program and software program source code for the software program;

extracting documentation data from said sources, wherein said extracting comprises analyzing a source for type and data format; and

selecting a corresponding one of a plurality of different input source plug- ins based on said analyzing to extract (refer to claim 1) the documentation data;

aggregating the extracted documentation data in a uniform format (refer to claim 1); and

transforming the uniformly formatted documentation data into one or more specified documentation sets (refer to claim 1) for the software program, wherein said transforming comprises:

selecting one transformer plug-in set corresponding to a specified output documentation format; and translating a portion of the uniformly formatted aggregate data into one or more elements of a software documentation set in the specified output documentation format (refer to claim 1)

Claussen does not explicitly disclose selecting *one of a plurality of transformer plug-in sets corresponding to a specified output documentation format* and using a *selected transformer* plug-in set for translating a portion of the uniformly formatted aggregate data into one or more elements of software documentation set in the specified output documentation format. But this plurality of transformer plug-ins set limitation has been addressed in claim 1.

As per claims 16, 18-19, refer to claims 2, 7, 5 respectively.

As per claim 21, Claussen discloses wherein said aggregating comprises: if a uniformly formatted aggregate input document specifies information not comprised in data extracted from the source, generating said information (refer to claim 11); and generating a uniformly formatted aggregate input document (see Fig. 4).

As per claims 23, 25, refer to claims 12, 14.

As per claim 26, Claussen discloses a computer accessible memory medium, storing program instructions, wherein the program instructions are computer-executable to:

input a plurality of sources related to a software program, wherein the plurality of sources comprises a plurality of different types of sources (refer to claim 1) and comprises one or more of a software library documentation file (refer to claim 1) for the software program and software program source code for the software program;

analyze the one or more source files to identify the type of each of the source files; extract information from the plurality of sources based on the type of the source (refer to claim 1);

analyze a source for type and data format (e.g. step 308 – Fig. 3; Fig. 4); and

select a corresponding input source plug-ins based on said analyzing to extract the information from the source (refer to claim 1);

aggregate the extracted information into a uniform format (refer to claim 1); and

transform the aggregated information into one or more specified sets of software documentation for the software program (refer to claim 1)

Claussen discloses using one transformer plug-in corresponding to a specified output documentation format corresponding to a specified output documentation format, and translating a portion of the uniformly formatted aggregate information into one or more elements of a software documentation set in the specified output documentation format (refer to claim 1)

Claussen does not explicitly disclose selecting one plug-ins from a plurality of input source plug-ins.

Nor does Claussen disclose selecting one of a plurality of transformer plug-in sets corresponding to a specified output documentation format and translating a portion of the uniformly formatted aggregate information into elements of a software documentation set in the specified output documentation format.

But these limitations have been rendered obvious in claim 1, respectively.

As per claims 27, 29, 30, 32, 34, 36, refer to claims 16, 18, 19, 21, 23, 25, respectively.

5. Claims 3, 17, 28 are rejected under 35 U.S.C. 103(a) as being unpatentable over Clausen et al., USPN: 6,732,330 and Murren et al, USPN: 7,000,185, and further in view of Yu et al, USPubN: 2004/0090458 (herein Yu).

As per claim 3, Claussen discloses wherein the software library documentation file but does not explicitly disclose a tag library descriptor (TLD). Similar to the process of using *taglibs* to process the elements of a DOM specific to a JSP format in Claussen, Yu teaches a descriptor provided to identify taglib URI listed as a tld.file (Yu: Fig. 5; Fig. 9). Based on the very descriptor being integral to the well-known *taglibs* for describing APIs for custom tags as set forth in Yu (col. 4 para 0046, pg. 4), it would have been obvious for one skill in the art at the time the invention was made to implement Claussen's use of taglibs in view determining of plug-

ins to extract information, so that the TLD (as described in Yu) be included in Claussen's process of identifying tag information and related APIs in order to invoke the proper plug-ins (based on that TL descriptor) in order to achieve the proper matching of resources and type of input, whereby enabling efficient usage of plug-ins as set forth in terms of plurality of plug-ins rendered obvious as per claim 1 (Claussen: plug-ins col. 4 lines 28-40)

As per claims 17 and 28, refer to claim 3.

6. Claims 13, 24, 35 are rejected under 35 U.S.C. 103(a) as being unpatentable over Clausen et al., USPN: 6,732,330 and Murren et al, USPN: 7,000,185, and further in view of Santos, USPN: 7, 107, 521 (herein Santos).

As per claim 13, Claussen discloses documentation set file being of any scripting language (e.g. language of choice – col. 2 lines 6-22) but does not explicitly disclose wherein the output software documentation sets comprise one or more portable document files (PDF). Analogous to Claussen in using a DOM and stylesheet to derive a target output file, Santos teaches conversion into a specific document format (Fig. 3, 5), wherein such format can be a PDF (col. 2 lines 26-38; col. 3 lines 17-31). Based on the capability to yield output document in a language of choice based on DOM and taglib in conjunction with XSLT approach by Claussen, it would have been obvious for one skill in the art at the time the invention was made to implement the generator of output documentation sets in Claussen so that DOM/XSL would be directed to generated PDF as set forth above, because this would fall into the endeavor by Claussen to select any target language for creating output documentation sets from the similar DOM, tag and stylesheet methodology which is also depicted in Santos

As per claims 24 and 35, refer to claim 13.

Response to Arguments

7. Applicant's arguments filed 8/29/08 have been fully considered but they are moot in light of the new grounds of rejection which have been necessitated by the Amendments.

Conclusion

8. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis Bullock can be reached on (571)272-3759.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Tuan A Vu/

Primary Examiner, Art Unit 2193

September 27, 2008